# Netta Shafir's Summer Project

Advisor: Prof. Yifat Prut

Aug-Sep 2021

## Introduction

Every type of neuron in the brain has it's own unique shape of spike, which is the action potential of the neuron when firing. When recording extracellular activity from the brain, the output is the sum of all spikes of all from the neurons around the electrode. The electrode has multiple tuning parameters that determines the number of neurons that the electrode can record effectively, while spikes from more distant neurons are considered as background noise. Another sources for noise is the neuron itself (as one neuron has some variance in the shape of it's spike), and also artifacts caused from the recording process itself (the animal moved, the stimuli that generated by the experimenter to activate some area in the brain, etc.). For a research relating to the correlation and the connections between couple of single neurons in the area that we are recording, there is a need to understand which neuron fired at which time point. This problem is known in the literature as **"Spike Sorting"**.

## The proposed solution

Most papers on this topic suggest an unsupervised solution. That is to say, a solution that uses prior knowledge regarding the general number of neurons being recorded (because this is determined by the tuning parameters of the electrode) but not the shape of their spike's waveforms. Such an unsupervised algorithms will find the waveforms of the single neurons (or the averaged waveform of each neuron), and the time series of the spike for each one. An example of this is the article we used in this project: "Sorting Overlapping Spike Waveforms from Electrode and Tetrode recordings"[1]. In our project, in addition to the extracellular recording itself, we used prior knowledge about the times of the spikes of the neurons that located in the area that was recorded, that was analysed manually by Prof. Prut (from now on it will be reffered to as the "true data"). It should be noted that this analysis was not flawless, so we considered it as a ground truth most of the time when we tried to test the program with it, but finally we also used the program to test this analysis as well.[2] Therefore, we should make some adjustments in the algorithm suggested in the article, and make it supervised instead of unsupervised. In the case of our article, it was not so difficult, because the stage of figuring out the waveforms of the single neurons occurs first, and is separated from the next stages of finding the spike times. So after cutting the head of the algorithms, the next stages are as follows (A full description, as well as the API for the python implementation, is written in the README file of the python code of this project):

1. **Pre-processing** - we tried the algorithm on simulated data first and on the true data afterwards, each one requires different processing:

   (a) In the simulator, we looked over a true extracellular record and picked manually a few spikes that their waveforms looked like spikes produced by single neurons, and chose them for the average spikes of our single neurons. Then we produced two Poisson processes for the simulated spikes, and fixed the refractory violation[3]. Then we created a single-unit record (the record that we would get if we would record a single neuron only). First, we creatied an array with the only value of the resting potential of a neuron (-70 volt in the literature). Then, every value in the appropriate Poisson process was an index in this array from it onwards we inserted the waveform of the appropriate

[1] Mokri Y, Salazar RF, Goodell B, Baker J, Gray CM and Yen S-C ,(2017) Sorting Overlapping Spike Waveforms from Electrode and Tetrode Recordings, Front. Neuroinform. .11:53 doi: 10.3389/fninf.2017.00053

[2] We assumed that most, if not all, the spikes that were identified in the true data were identified correctly, but we suspected that there are more spikes that were not identified because they overlap with another spikes. So the mean spike's waveform of each neuron was considered by us to be exact, and therefor could be used to find another hidden spike in the record that was not identified in Prof. Prut analysis.

[3] We assumed that there must be at least 2 millisecond between every two spikes, considering the refractory stage the neuron is having after every spike. So we removed from the Poisson process spikes that violated this restriction.

spike, multiplied by a noise factor that was taken from a gaussian random variable (which it's standard deviation had received as an input).[4]. Summing the two single-unit record gave us the simulated multi-unit record[5].

(b) In the true data case, we got the multi-unit record and the spikes times as an input. Therefore we take all spikes of every single neuron, and considered the average waveform as the noiseless shape of this neuron, for the average an unbiased estimator for the expected value was determined.

(c) In both cases, the main function gets as an input the multi-unit record, the waveforms of the single neurons and their spike times (actually spike indices). All of the waveforms, as well as the waveforms we program, finds in the record later on, have the same length of 2 milliseconds and are aligned by their positive peak.

2. **Spike Extraction** - the program finds all the spikes in the extracellular record, that their positive peak is reaching some threshold. The threshold was 20,000 volt for the simulated data, and 8,000 volt for the true data. We called these spikes **overlapping spikes**, because we assumed that most of them are sums of spikes coming from different single neurons.

3. **Generation of Synthetic waveforms** - we took every two distinct neurons in the data, shifted their waveforms relative to each other, and then linearly added them together point by point to create a synthetic overlap waveform corresponding to the shift. We called these spikes **synthetic spikes.**

4. **Clustering** - we took all the spikes we had so far (the single-unit spikes, the overlapping and the synthetic spikes) and classified them into different clusters[6]. Then, for each cluster we examined all the overlapping spikes, and found the synthetic spike (or the single-unit spike) in it's cluster that has the highest Pearson's correlation with it[7]. This synthetic waveform was then identified with the overlap waveform, and the two single-unit spikes that created it are considered to create the overlap waveform. This stage is illustrated in this diagram from the original paper:

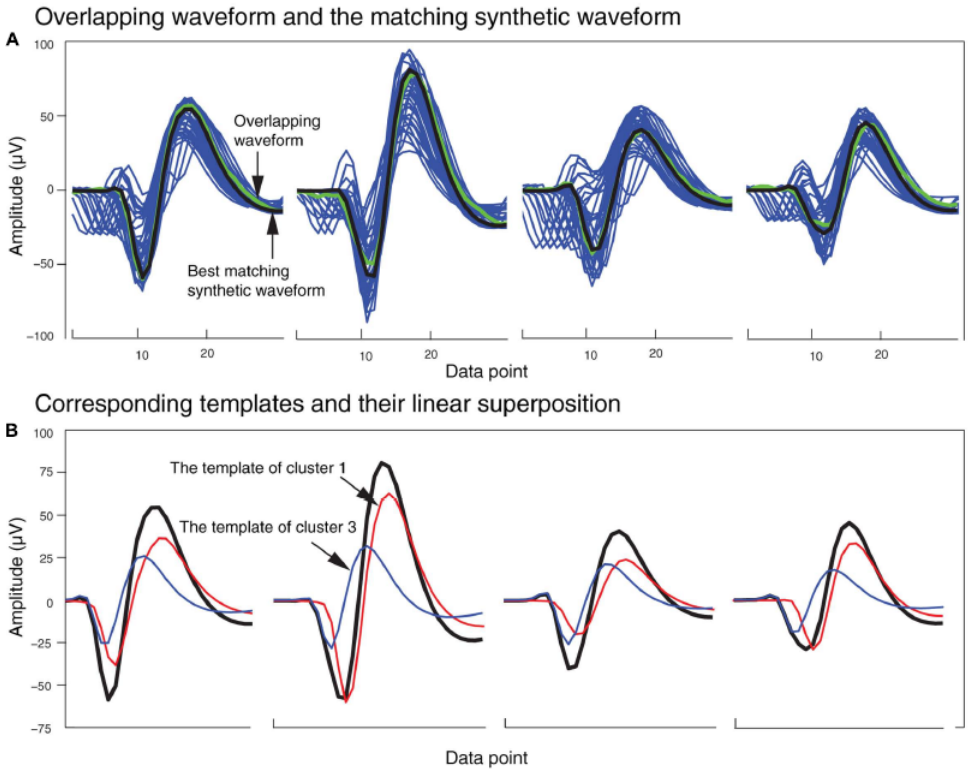



**FIGURE 5 |** Matching of synthetic and overlapping waveforms for the tetrode data recording shown in **Figures 2**, **4**. **(A)** The synthetic waveforms (blue) clustered with one overlapping waveform (green) are shown with the best-matched synthetic waveform highlighted in black. **(B)** The template waveforms (blue and red) that generated the synthetic waveform (black) shown in **(A)**. The cluster numbers refer to the cluster numbers in **Figure 2B**.

[4]Instead of this, at the start we added a background noise taken from gaussian distribution (see Results), but later on in the project we found this noise less interesting to be examined, because most of it was ignored anyway for being under the volt threshold of the program (see section .(2

[5]After checking the algorithm on two simulated single units, we decided to try it on the true data and not on a more complicated simulated data.

[6]The biggest problem in any clustering method is to determine the desirabel amount of clusters. The team of the original paper used a package named KlustaKwik which works iteratively to determine the best amount of clusters, but when we tried it we wasn't satisfied from it's results. So, we used the K-means algorithms and used the heuristic of taking 15 clusters.

[7] At some point we determined some threshold the correlation is needed to exceed for the two waveforms will be considered as "similar".

This stage ends the algorithm. After this, we could reconstruct an estimated single-unit record with the single-unit waveforms we matched with the overlapping waveform, and the spikes times of these overlapping waveforms.

5. **Loss evaluation -** this is an optional stage to evaluate the mistake in the algorithm estimation of the single-units. We calculated FPR (false positive rate) and FNR (false negative rate) using that method: For the FPR, for every single-unit we took an array with the length of our record that have `1` in a slot of a spike's peak in the true single-unit and `0` everywhere else, and then draw a gaussian around every peak, using the convolution formula[8]. If we mark this array with "$src$", and the set of peak times (indices corresponding to the length of $src$) of the spikes in the output single-unit of the program with "$I$", the FPR then was calculated with this formula:

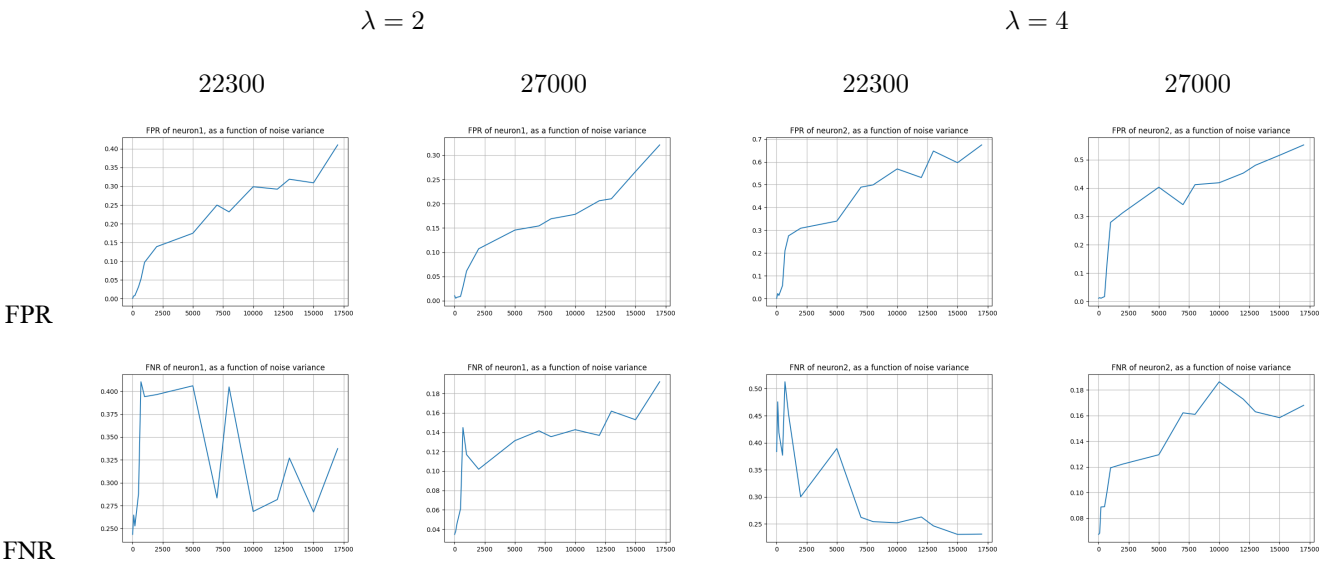
$$FPR = \frac{\sum_{i \in I} 1 - src\,[i]}{|I|} = 1 - \frac{\sum_{i \in I} src\,[i]}{|I|}$$

Therefore, if every index of a peak time that appears in a single-unit of the output of the program is also appears in the single-unit of the input of the program (the true data or the simulated data), the FPR is `0`. The FNR is calculated with the exact same way, with the output and the input peaks alternates.

# Results

### Loss as a function of noise in the simulated data

First, we checked the FPR and the FNR as a function of the variance of the noise in the record, when the noise we checked here was background noise that has taken from a gaussian distribution. We checked it twice: first on a pair neurons whose waveforms has relatively high correlation between them (around `0.95`) and then on a pair of a neuron and an artifact (a record of a stimulation which is a part of the experiment) whose waveforms has low correlation between them (around `0.12`). For each of these pair and for each value of variance we checked, we ran the programs `10` times and compute the average FPR and FNR, and we repeat this process twice: The first time one neuron had a parameter of $\lambda = 2^9$ to it's Poisson process and the other got $\lambda = 4$, and the next time the opposite. Overall, we got `16` graphs of loss (FPR or FNR as a function of the standard deviation of the noise in the spike's shape). For the high correlated pair, these are the results we got[10]:

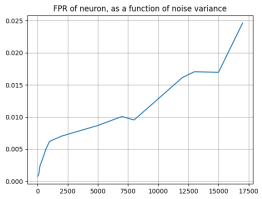
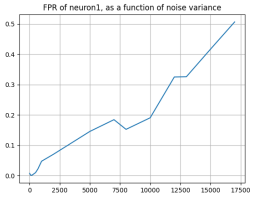
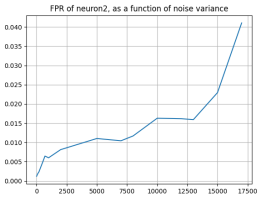
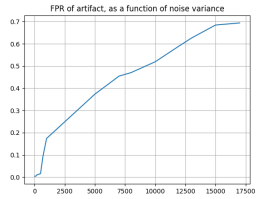
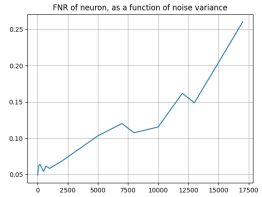
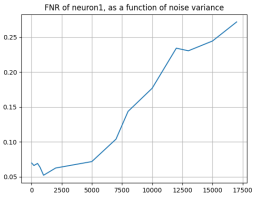
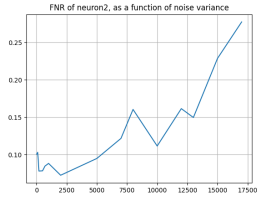
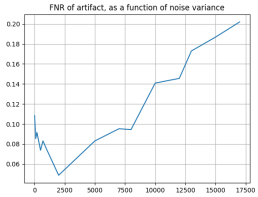



For the low-correlated pair, these are the results we got:

[8] It actually was a curve with a shape of a gaussian, that it's peak has a value of `1`. The width of that curve was reflected the tolerance for nearby spikes in the estimated single-unit record (the output of the program), and could be determined with an input parameter of the program.

[9] Which means that the expected value of the time passes between two neighbor spikes is `2` milliseconds.

[10] The numbers 22300, 27000 etc. are for recognition of a neuron, and named after the positive peak value of its spike's waveform.

| | $\lambda = 2$ | | $\lambda = 4$ | |
|---|---|---|---|---|
| | 22200 | Artifact1 | 22200 | Artifact1 |
| FPR |  |  |  |  |
| FNR |  |  |  |  |

We can see that the general trend of (almost) all the graphs is increase of the error (FPR or FNR), as we would expect. From a glance at these graphs, we could conclude couple of insights:

1. For both pairs (i.e. same for the high correlated pair as well as the low correlated pair), the FPR is generally higher when $\lambda$ is higher.
2. For both lambda values, in the low correlated pair the FPR is generally lower than the FNR, and the opposite is true for the high correlated
3. In the high correlated pair and for both lambda values, the FNR of the 22300 neuron has a downward trend, unlike the general behavior we would expect.

Nevertheless, this data analysis is indeed simplified, and future research is required to confirm the results.

**FNR as a function of overlap in the simulated data**

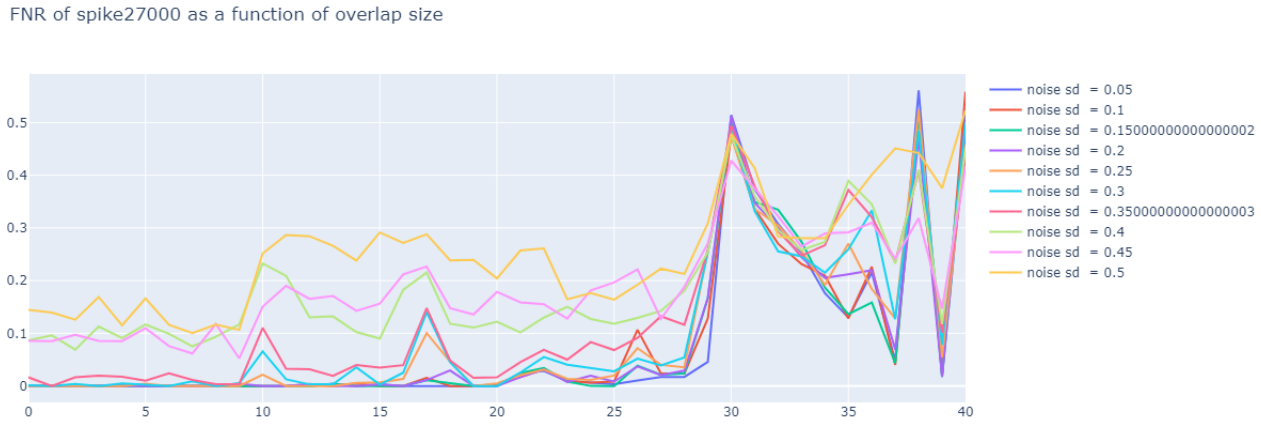
We were interested in the success of the algorithm in spikes that are actually a sum of two or more overlapped spikes. That is because the existing methods that are based on simple clustering[11] are already quite successful analysing spikes that don't overlap another spikes. So, for a single neuron, we checked the FNR as a function of data points of overlap of it's spike with another neuron's spike (we were less interested in the FPR of this case). We measured the FNR of the high correlated pair[12], for a different noise with different values os standard deviation[13]. These were the results we got (the horizontal axis counts data points of overlap between the two spikes):

[11] For example programs like Plexon that are very common in research.

[12] Neuron 22300 has a parameter of $\lambda = 2$ in the Poisson process, and neuron 27000 has $\lambda = 4$.

[13] The noise here was a factor multiplied in the spikes' waveform, and not the background noise as before.


It can be observed that the general behavior is as expected: the more noise, the more error. An interesting behavior is this: Up to 32 points of overlap, the error indeed increase as we would expect, but from 32 to 40 the error actually decreases.




As for the 27000 neurons, here the behavior is even more unclear in the range of 30-40 points of overlap.

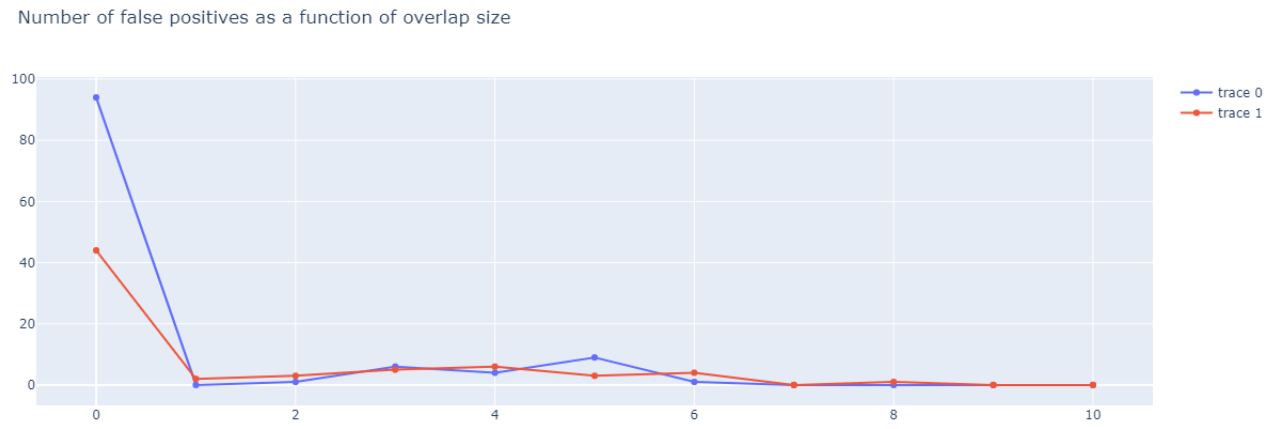
**False Positive as a function of overlap in the true data**

Then we moved to examine the algorithm's performances on the true data[14]. The results of the algorithm on the true data are as follows:

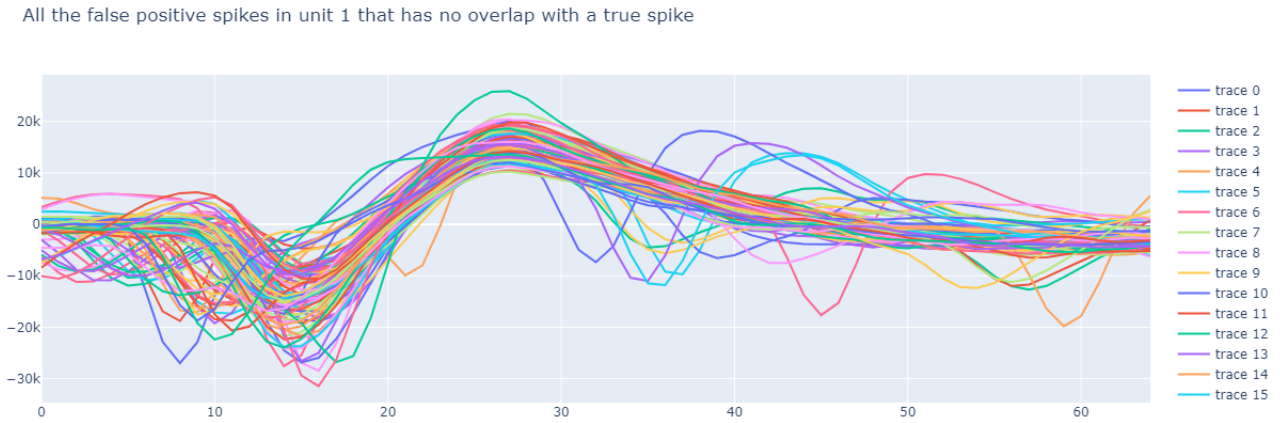
[14]The true data had been taken from a file named "data4adi". This file contains a struct named "dall", and we used the first element from it that included an extracellular record, and spike times of 4 neurons manually analysed by Prof. Prut.

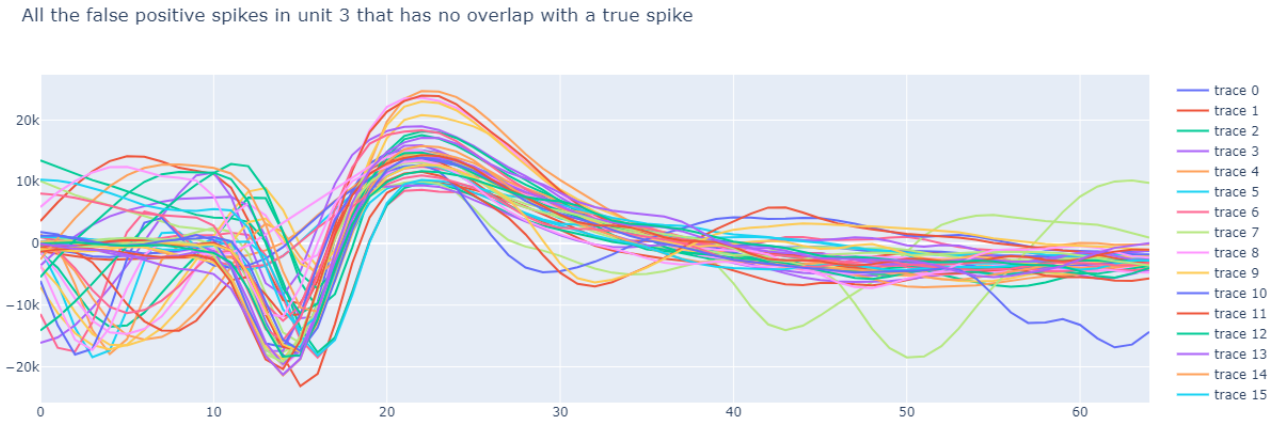
```
Results:
unit 1:
    original spikes 2477, estimated 2074
    FPR 0.10732869129729873, FNR 0.25256346618921127
unit 2:
    original spikes 1927, estimated 491
    FPR 0.9472663335093566, FNR 0.9865634508319313
unit 3:
    original spikes 1075, estimated 1042
    FPR 0.11274431618429026, FNR 0.13998100229212793
unit 4:
    original spikes 120, estimated 483
    FPR 1.0, FNR 1.0
```

As before, we were interested in the influence of the overlap of a spike with the spike of another neuron on the capability of the algorithm in recognizing it. Here, we wanted to observe the capability of the algorithms to identify spikes that were not observed in the manual analysis (probably because they overlapped another spikes). Therefore, we checked the true positives of the algorithm on the true data, as a function of overlap **with spikes that had been recognized in the manual analysis**. We checked it on two neurons out of the four in total that the FPR and FNR of the algorithms on them was relatively small (around 0.15 on average). These were the results (the horizontal axis counts tens of percent):

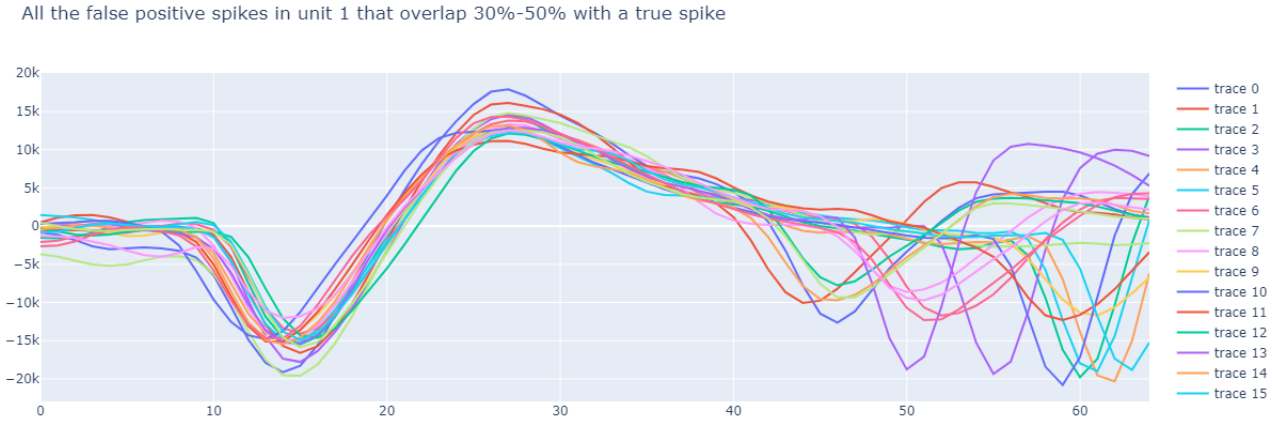


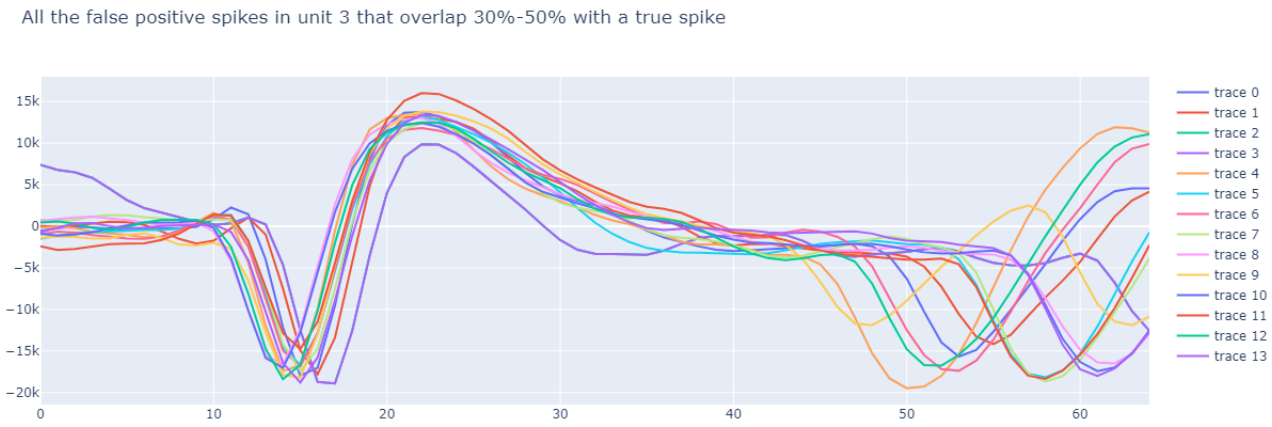

It can be observed that the highest amount of FP spikes did not overlap another spike from the manual analysis. Another rise in the graph is observed around 30%-50% of overlapping. When plotting those spikes (with no overlap), we get the following graph:




One can see that the shape of most of the spikes is the shape of a sum of two overlapped spikes, which means that the FP spikes did actually overlapped with another spikes that hadn't been discovered in the manual analysis. The same picture is seen as well in the FP spikes that overlapped by 30%-50% percent with another spike in the true data:

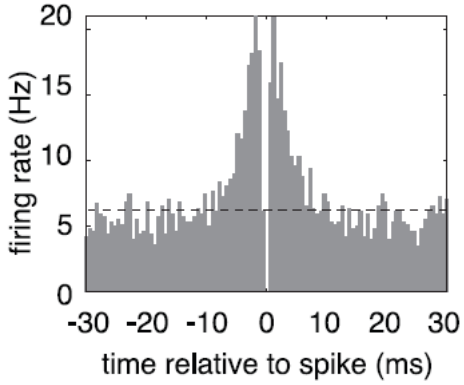





One can observe that all the spikes that had been classified to one of the clusters (unit 1 or unit 3) are quite similar to each other in their main part, so it is quite reasonable to assume that most of these spikes were indeed fired by the same neuron, and that the program classified them properly. These results imply that the manual methods we have today do not give satisfactory results.

**Cross-correlation**

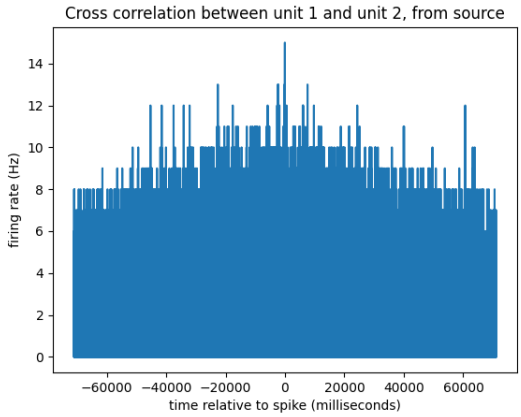
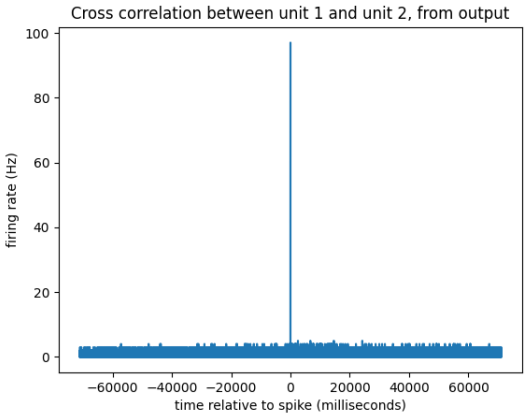
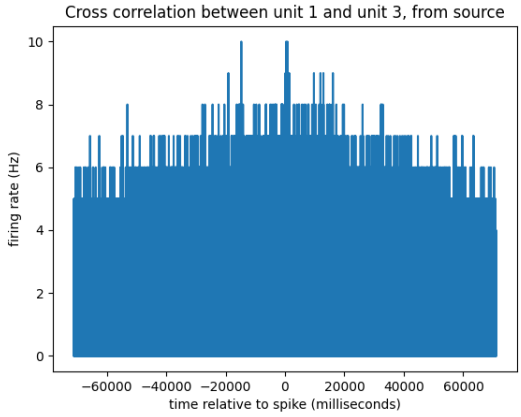
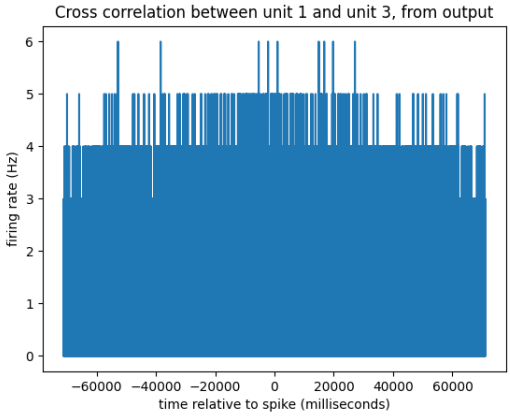
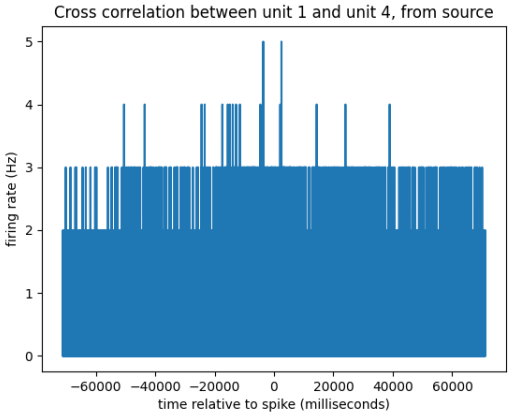
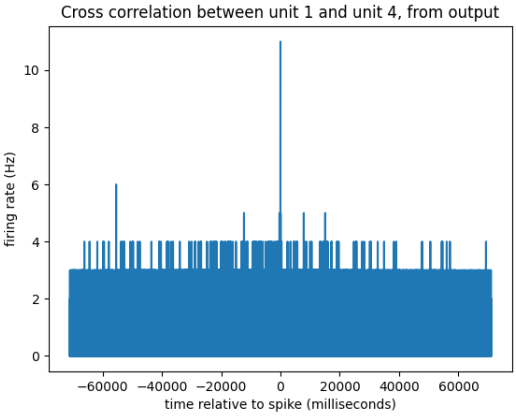
Finally, we asked to compare the cross-correlation between the different neurons in the true data, to the cross-correlation between the same pairs of neurons in the output of the program. The CCF (cross -correlation function) of two neurons can reflect the relation between them. For example, a CCF of two neurons whose firing is highly correlated in time (e.g. that have synapse with third neuron), and therefor are correlated, is as follows:

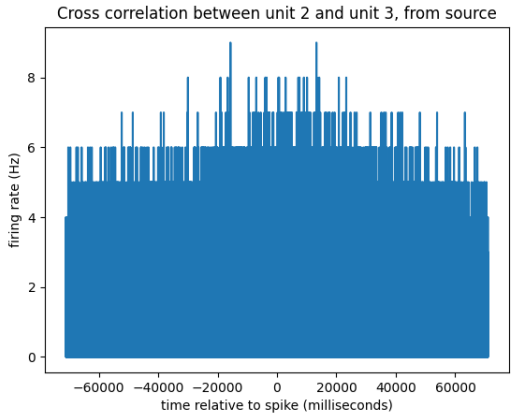
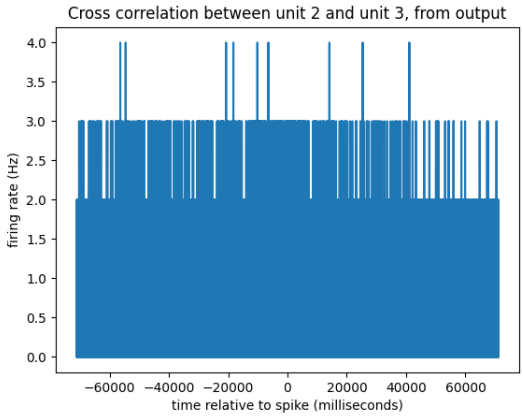
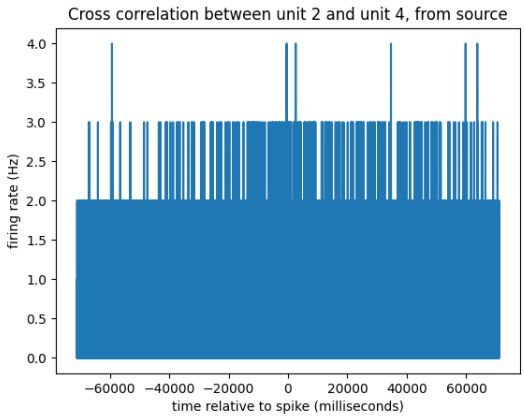
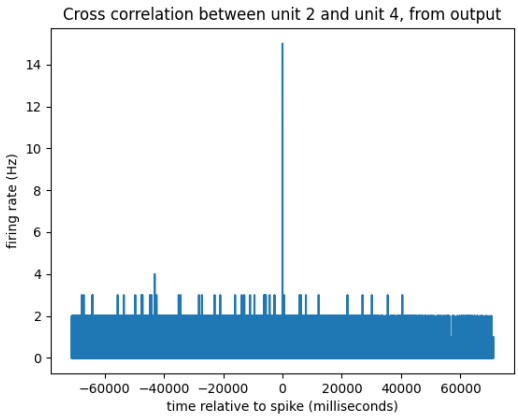
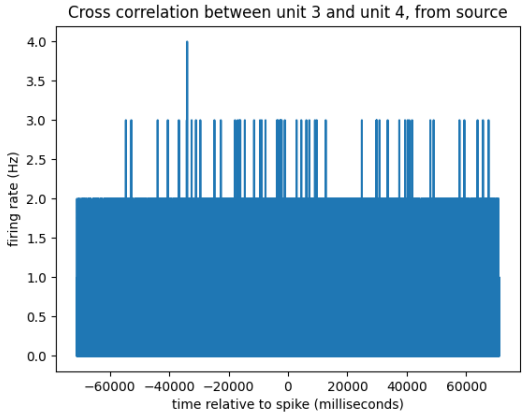
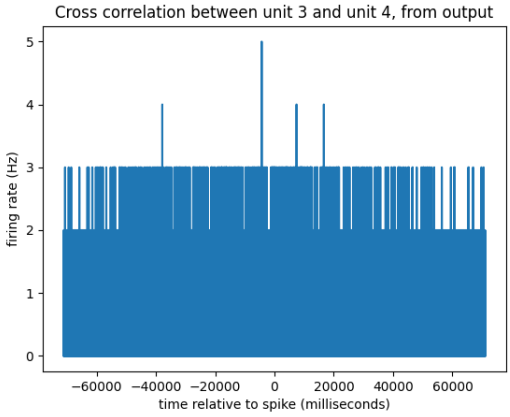



The frequency of the spikes of one neuron increases as it gets closer to the spike of the other neuron. The sharp notch in the curve around $\pm 1$ is caused by the clustering problem: when spikes are overlapped, the traditional methods of spike-sorting do not succeed in recognize them properly. In such cases, we would expect that the output of a successful spike-sorting program will create a CCF with this notch full.

After running the algorithm on the true data, and computing the CCF between all combinations of neurons for the spikes in true data[15], these were the results we got:

[15]The CCF between neuron 1 and neuron 2 is a mirror picture to the CCF between neuron 2 and neuron 1.

| Input | Output |
| --- | --- |
|  |  |
|  |  |
|  |  |

| Input | Output |
| --- | --- |
|  |  |
|  |  |
|  |  |

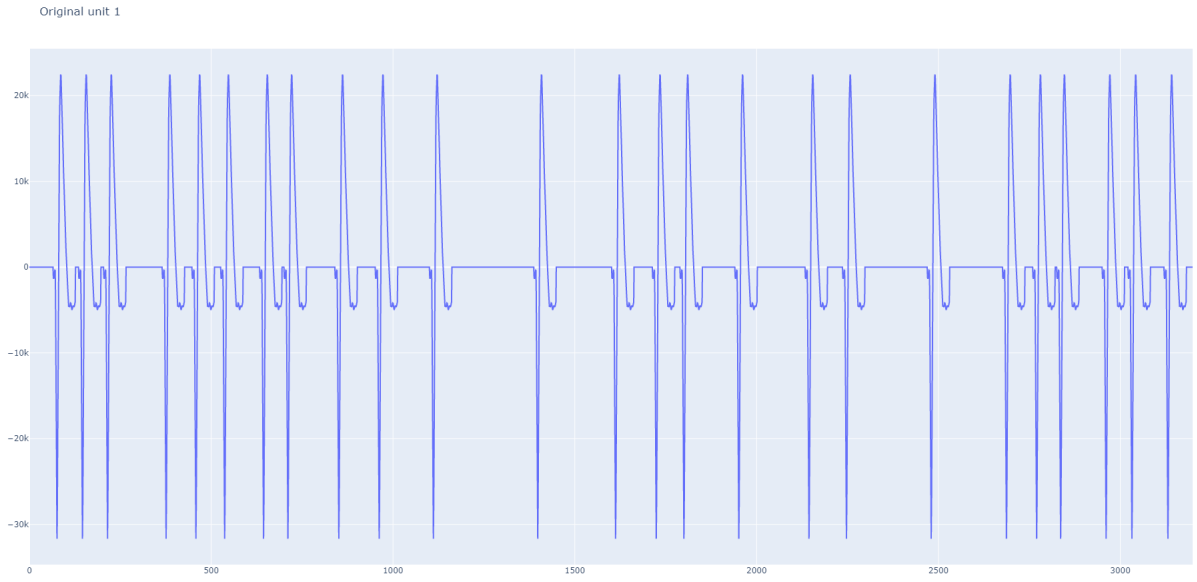


## Summary and future improvements

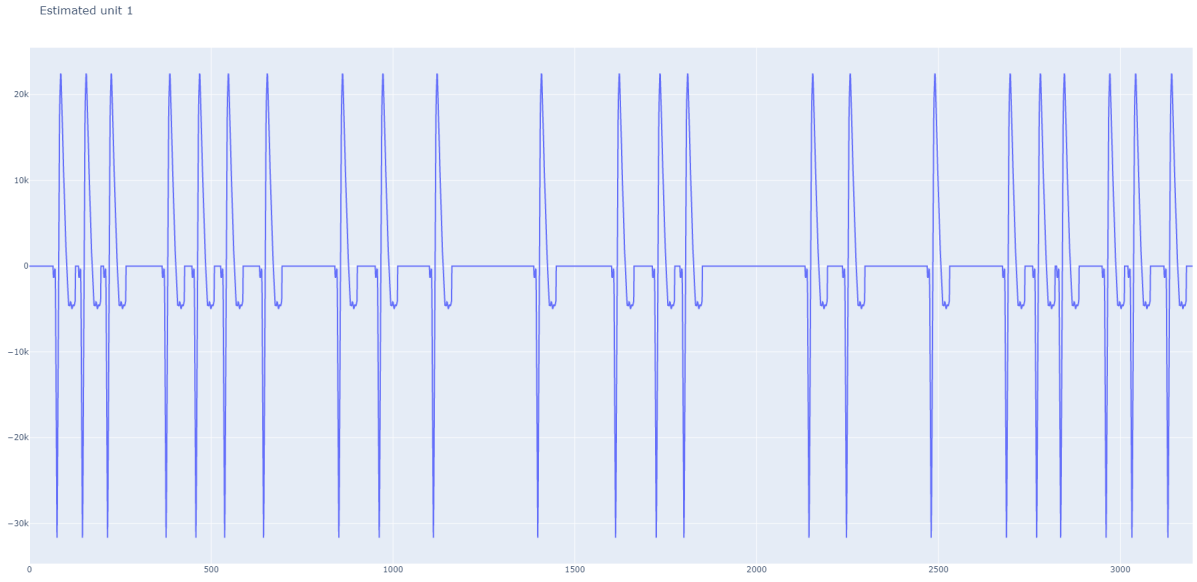
In this project we tried to offer a partial solution for the Spike-Sorting problem. Our solution was based on the algorithm suggested in the article by Mokri et al. from "Frontier in Neuroinformatics": we implemented a supervised version of this algorithm that gets the shapes of the neurons that are recorded (and optionally their spikes' times), instead of figure them out from the record as in the original algorithm.

Our algorithm gave nice results on the simulated data and partially on th true data. For example, these are the results of the program on record generated by the simulator of the highly-correlated waveforms:

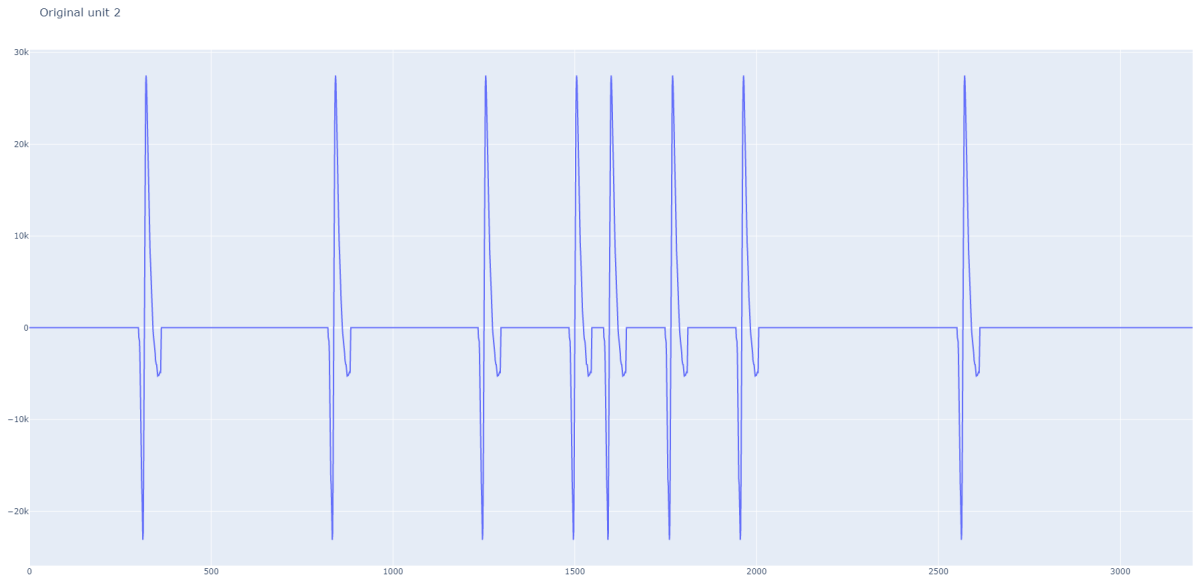
**Original unit 1:**



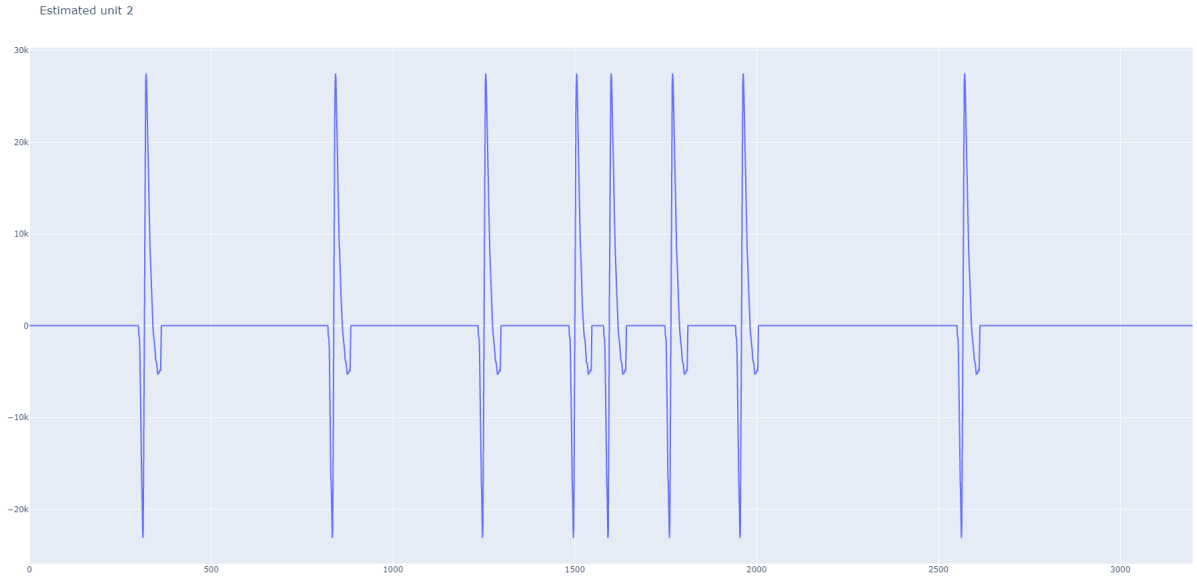

**Estimated unit 1:**




**Original unit 2:**




**Estimated unit 2:**




```
Results:
unit 1:
    original spikes 25, estimated 23
    FPR 0.0, FNR 0.08
unit 2:
    original spikes 8, estimated 8
    FPR 0.004951829254143406, FNR 0.004951829254178072
Correlation = 0.9435228405200025
```

I think that the biggest change that can improve the performances of the algorithm is another clustering method. As mentioned above, the current clustering method the algorithm use is based on heuristics only, and we didn't tried any other method because it wasn't the focus in this project. The core of this algorithm is the matching between the overlap spike and

the synthetic spike, and this stage happening inside the cluster that was created in the clustering stage. A better clustering method, that pick the proper amount of clusters automatically[16], will theoretically give a more accurate algorithm.

[16] As in KlustaKwik in the original paper.